KERJA PRAKTIK - IF184801

Studi Perbandingan Literatur Server Database Dalam Load & Balancing Charging System Cluster Sebagai Sarana Transaksi Pembayaran Jalan Tol Otomatis

Telkom Landmark Tower, Jl. Gatot Subroto No.Kav. 52, RT.1/RW.1, West Kuningan, Mampang Prapatan, South Jakarta City, Jakarta 12710

Periode: 13 Agustus 2020 - 11 September 2020

Oleh:
Budiman Akbar Radhiansyah 05111740000179

Pembimbing Jurusan
Siska Arifiani, S.Kom., M.Kom.
Pembimbing Lapangan
Masrizal Frisnanto

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

KERJA PRAKTIK - IF 184801

Studi Perbandingan Literatur Server Database Dalam Load & Balancing Charging System Cluster Sebagai Sarana Transaksi Pembayaran Jalan Tol Otomatis

Telkom Landmark Tower, Jl. Gatot Subroto No.Kav. 52, RT.1/RW.1, West Kuningan, Mampang Prapatan, South Jakarta City, Jakarta 12710

Periode 13 Agustus 2020 - 11 September 2020

Oleh:

Budiman Akbar Radhiansyah 05111740000179

Pembimbing Jurusan

Siska Arifiani, S.Kom., M.Kom.

Pembimbing Lapangan

Masrizal Frisnanto

DEPARTEMEN TEKNIK INFORMATIKA

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya 2020

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN
KERJA PRAKTIK

Perbandingan Server Database Dalam Load & Balancing Charging System Cluster Sebagai Sarana Transaksi Pembayaran Jalan Tol Otomatis Menggunakan Apache Jmeter

Oleh:

Budiman Akbar Radhiansyah **05111740000179**

(Mahasiswa)

Disetujui Oleh Pembimbing Kerja Praktik:

1. Siska Arifiani, S.Kom., M.Kom.

(Pembimbing Departemen)

2. Masrizal Frisnanto

Digitally signed by 896412
DN: c=ID, st=DKI Jakarta, l=Jakarta Selatan, o=PT Finnet Indonesia, ou=INFRA, cn=896412, email=frisnanto@finnet-indonesia.com
Date: 2021.02.05 20:07:28 +07'00'

(Pembimbing Lapangan)

[Halaman ini sengaja dikosongkan]

Studi Perbandingan Literatur Server Database Dalam Load & Balancing Charging System Cluster Sebagai Sarana Transaksi Pembayaran Jalan Tol Otomatis

Nama Mahasiswa	: Budiman Akbar Radhiansyah (05111740000179)
Departemen	: Teknik Informatika FTEIC - ITS
Pembimbing Departemen	: Siska Arifiani, S.Kom., M.Kom.
Pembimbing Lapangan	: Masrizal Frisnanto

ABSTRAK

Dengan semakin maraknya pengguna jalan tol di Indonesia khususnya wilayah Jabodetabek pada tahun 2020, diiringi dengan Pemerintahan Indonesia memperketat social distancing maupun physical distancing pada setiap sudut wilayah Jabodetabek. Pada pertengahan tahun, PT. Finnet Indonesia berencana merancang jalan tol hand-free untuk keperluan masyarakat Jabodetabek serta nantinya akan dibawa ke penghujung nusantara. Karena faktor tersebut, Pemerintahan membutuhkan suatu sistem berbasis hand-free yang dapat memudahkan serta melancarkan jalan tol di daerah tersebut.

Sistem ini dibuat dengan menggunakan Database Server seperti Cassandra, MongoDB, SQL. Dalam modul yang dibahas, pengguna dapat dengan mudah melakukan keluar-masuk jalan tol tapi untuk kali ini hanya akan dibahas mengenai bagian Testing Database Server yang digunakan.

Kata kunci: Database Server, PT. Finnet Indonesia, Jalan Tol otomatis, Testing

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT karena berkat rahmat dan lindungan-Nya penulis dapat melaksanakan salah satu kewajiban kami sebagai mahasiswa Departemen Teknik Informatika, yakni Kerja Praktik (KP).

Penulis menyadari masih ada kekurangan baik dalam pelaksanaan KP maupun penyusunan buku laporan ini. Namun, penulis berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi. Penulis mengharapkan kritik dan saran yang membangun untuk kesempurnaan buku laporan KP ini.

Melalui buku ini, penulis juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu, baik langsung maupun tidak langsung, dalam pelaksanaan KP hingga penyusunan laporan. Orang-orang tersebut antara lain adalah:

1. Kedua orang tua penulis.
2. Ibu Siska Arifiani, S.Kom., M.Kom. selaku dosen pembimbing Kerja Praktik.
3. Bapak Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D. selaku Koordinator Kerja Praktik.
4. Bapak Masrizal Frisnanto selaku pembimbing lapangan selama kerja praktik yang telah memberikan bimbingan serta ilmunya kepada penulis.
5. Teman-teman penulis yang senantiasa memberikan semangat ketika penulis melaksanakan KP.

Jakarta, Agustus 2020

Penulis

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

BAB I
PENDAHULUAN

1.1. Latar Belakang

Pandemi yang sedang dialami oleh Indonesia, yakni merebaknya virus COVID-19, menyebabkan diberlakukannya pembatasan social berskala besar dalam rangka percepatan penanganan penyebaran virus. Pembatasan sosial ini berpengaruh pada penggunaan jalan tol yang marak digunakan masyarakat indonesia, diantaranya adalah warga Jabodetabek. KP ini mengusulkan salah satu solusi masalah ini yaitu pengimplementasian pembayaran jalan tol yang bersifat non-fisik atau *handsfree.*

Saat ini dunia telah berkembang menjadi era digital. Dimana semua layanan dapat disajikan secara *easy-to-use*, dapat dimanfaatkan untuk mempermudah manusia dalam melakukan layanan tersebut. Langkah awal dalam pembatasan sosial tersebut adalah dengan mengusahakan seluruh komunikasi maupun layanan yang ada dapat diakses secara non-fisik. hal ini lah bisa disebut merupakan layanan yang perlu difasilitasi dalam pelaksanaannya sekarang ini. Sistem ini dibuat untuk memfasilitasi masyarakat Indonesia. Pembayaran Tol bersifat non-fisik (*handsfree*), sehingga peserta dapat melakukan transaksi tol tanpa membuka kaca mobil masing-masing serta menggunakan perangkat kartu tol elektronik (E-toll) sehingga jalan tol dapat berlangsung lancar dengan menaati peraturan pembatasan sosial yang berlaku. Pada setiap server terdapat pengawas yang akan membantu langsung dengan pengguna melalui hub yang tersedia, serta akan memantau perkembangan pengguna dalam melakukan transaksi. Selain pengawas, terdapat otoritas admin yang dapat melakukan *maintenance dalam mesin server*. Tim *maintenance* ini dapat melakukan testing yang telah disiapkan ke dalam sistem melalui *interface*, dan melakukan asuransi kualitas pada sub-bagian server dari paket-paket yang diterima dari transaksi. Mesin toll menyediakan fitur untuk mendeteksi mobil-mobil pengguna di tiap sub-bagian jalan, dimana admin dapat mengakses fitur tersebut.

Dalam KP ini disajikan kajian review beberapa literatur tentang teknologi yang akan digunakan sebagai dasar kajian pustaka dalam mengimplementasikan metode jalan tol bersifat non-fisik secara spesifik dan mendalam.

1.1. Tujuan

Tujuan Kerja Praktik ini adalah untuk menyelesaikan kewajiban kuliah kerja praktik di Institut Teknologi Sepuluh Nopember dengan beban 2 SKS. Selain itu juga untuk memenuhi kebutuhan yang diperlukan oleh PT. Finnet indonesia dengan melakukan perbandingan antar database server. database ini menitikberatkan pada t*raffic user*.

Tujuan dari pengimplementasian aplikasi tersebut antara lain:

1. Melakukan penelitian secara mendasar mengenai database-database yang ada seperti Cassandra, MongoDB, serta MySQL
2. Mempelajari alat testing yang tepat dalam menentukan *traffic user* seperti Jmeter dan SoapUI
3. Melakukan testing terhadap *database* tepat untuk *traffic user* yang tinggi.

1.2. Manfaat

Manfaat yang kami peroleh dalam pelaksanaan kerja praktik ini adalah:

1. Pengalaman dalam lingkungan kerja yang sebenarnya.
2. Meningkatkan kemampuan kerjasama dalam tim.
3. Meningkatkan kemampuan mempelajari sistem baru.
4. Meningkatkan kemampuan berkomunikasi dengan orang baru.

Manfaat yang dapat diperoleh dengan adanya Testing Perbandingan Database ini adalah:

1. Memudahkan admin dalam pengelolaan akun, skenario traffic, penerimaan paket transaksi, serta data testing kedepannya.
2. Memudahkan peserta dalam melakukan transaksi tanpa halangan (*Lag-free*).
3. Memudahkan pelacakan data yang diterima pada database mana.

1.4. Rumusan Masalah

Berikut ini rumusan masalah dalam pelaksanaan kerja praktik perbandingan

database server:

1. Bagaimana cara menentukan poin banding antar db server?
2. Bagaimana solusi efektif dalam penentuan db server yang dapat mempermudah transaksi dan pengelolaan traffic user?

1.5. Lokasi dan Waktu Kerja Praktik

Kerja praktik ini dilaksanakan pada waktu dan tempat sebagai berikut:

Lokasi	: Work From Home (di rumah masing-masing)
Waktu	: 13 Agustus 2020 – 11 September 2020
Hari Kerja	: Senin - Jumat
Jam Kerja	: 08.00 WIB - 17.00 WIB

1.6. Metodologi Kerja Praktik

Tahapan pengerjaan kerja praktik dapat dijabarkan sebagai berikut:

1. Perumusan Masalah

Untuk mengetahui domain dan fungsionalitas, dijelaskan secara rinci bagaimana sistem yang harus dibuat. Penjelasan oleh Pembimbing Lapangan kerja praktik menghasilkan beberapa catatan mengenai gambaran secara garis besar tentang sistem atau server apa saja yang harus dipelajari untuk kedepannya digunakan pada sistem. Setelah mendapatkan gambaran sistem, terdapat diskusi lebih lanjut dengan diputuskan bahwa aplikasi untuk melakukan testing ialah Jmeter atau SoapUI. Selanjutnya, programmer dapat menggunakan tools lainnya sebagai pendukung kelengkapan fitur yang dibuat. Hal ini didasari oleh kompatibilitas serta data traffic dari sistem tol sebelumnya di daerah Jabodetabek.

2. Studi Literatur

Setelah ditentukan sistem, *database*, serta *tools* tambahan yang akan digunakan, dilakukan studi literatur mengenai cara implementasinya dalam membangun sistem sesuai yang dibutuhkan. Pada tahap ini dilakukan proses pencarian, pembelajaran, pengumpulan dan pemahaman informasi serta literatur yang berkaitan untuk membantu dalam implementasi sistem ini. Informasi bisa didapat dari internet untuk istilah-istilah umum yang digunakan dalam mengimplementasikan suatu sistem informasi.

3. Hasil Tinjauan Literatur

Langkah ini meliputi penjelasan awal tentang sistem. Bagaimana cara kerja sistem dengan skenario tertentu. Dari penjelasan awal telah didapatkan beberapa kebutuhan fungsional dan non-fungsional secara garis besar. Kemudian dilanjutkan dengan memperjelas spesifikasi kebutuhan-kebutuhan tersebut. Dibuatlah sebuah diagram kasus penggunaan yang mewakili skenario-skenario untuk penggunaan sistem aplikasi. Dilanjutkan dengan diskusi bersama pembimbing lapangan untuk mengetahui kebutuhan-kebutuhan tersebut telah tepat atau tidak.

1.7. Sistematika Laporan

Laporan KP ini terdiri dari tujuh bab dengan rincian sebagai berikut:

1. Bab I Pendahuluan

Pada bab ini dijelaskan tentang latar belakang permasalahan, tujuan, waktu pelaksanaan, serta sistematika pengerjaan KP dan juga penulisan laporan KP.

2. Bab II Profil Perusahaan

Pada bab ini, dijelaskan secara rinci tentang profil Direktorat Pascasarjana dan Pengembangan Akademik ITS.

3. Bab III Tinjauan Pustaka

Pada bab ini, dijelaskan mengenai konsep-konsep pembuatan model, teknologi, tinjauan pustaka dan literatur yang digunakan dalam penyelesaian KP.

4. Bab IV Hasil Tinjauan Literatur

Pada bab ini, dijelaskan hasil pembelajaran atau analisis terhadap apa saja yang diperlukan dan harus diperhatikan dalam pengembangan aplikasi yang dikerjakan selama KP.

5. Bab VII Kesimpulan dan Saran

Pada bab ini, dipaparkan kesimpulan yang dapat diambil dan juga saran selama pengerjaan KP.

[Halaman ini sengaja dikosongkan]

BAB II
PROFIL INSTANSI

2.1. Profil Instansi

PT Finnet Indonesia merupakan salah satu anak perusahaan dari PT Telekomunikasi Indonesia (Persero). Didirikan pada tanggal 31 Oktober 2005 dan mulai beroperasi tanggal 26 Januari 2006. PT Finnet Indonesia berfokus pada penyediaan infrastruktur teknologi informasi, aplikasi dan konten untuk melayani kebutuhan sistem informasi, dan transaksi keuangan bagi industri perbankan, hingga jasa keuangan lainnya. Dengan kepemilikan saham. PT Telekomunikasi Indonesia, Tbk. (melalui PT Multimedia Nusantara) sebesar 60% dan Yayasan Kesejahteraan Karyawan Bank Indonesia (melalui PT. Mekar Prana Indah) sebesar 40%, Finnet telah sukses mengembangkan perluasan layanan di bidang transaksi keuangan yang beragam, sesuai dengan kemajuan teknologi transaksi pembayaran. Pada tahun 2018, PT Finnet Indonesia telah terhubung dengan 122 Biller, 90 Bank, 100 ribu outlet, 800 online merchant, 7 negara dan membukukan 1,2 Miliar transaksi (per tahun 2018) dengan nilai transaksi sebanyak Rp.132 Triliun rupiah. PT Finnet Indonesia terus berinovasi dengan perkembangan kemajuan teknologi dan keperluan transaksi pembayaran untuk mempermudah masyarakat dalam melakukan berbagai transaksi pembayaran secara elektronik, yang memiliki cakupan yang luas, aman dan mudah.

2.2. Visi dan Misi

1. Visi

Menjadi Akselerator Utama Inklusi Keuangan Indonesia

2. Misi

- Menjadi bagian integral dari pemerintah untuk program peningkatan pendapatan pajak baik baik pusat maupun daerah melalui digitalisasi layanan keuangan

- Menjadi bagian integral dari pemerintah sebagai pendukung utama dalam penyaluran dana bantuan sosial maupun permodalan usaha kecil dan menengah melalui teknologi keuangan digital

- Menjadi mitra teknologi layanan keuangan digital yang utama bagi perkembangan perbankan konvensional

- Menjadi kanal utama bagi penyaluran layanan keuangan beragam (multi financial service) bagi masyarakat kelas menengah, kecil, maupun komunitas

- Menjadi pilihan Utama bagi sektor utama usaha swasta sebagai mitra penyedia layanan keuangan digital yang mendukung supply chain management

2.3. Portofolio Instansi

- Golden Stevie Winner

  Fintech Bill Payment on E-Commerce

  (Payment aggregator,Switching, Multi Biller)

- Silver Winner

  BUMN Marketeers Awards 2018 Silver Winner Kategori Anak Perusahaan Promising Company In Strategy

- Silver Stevie Winner

  Fintech Bill Payment on a single click

- Silver Stevie Winner

  Fintech Loyalty Platform

  (E-money, Electronic Ticketing, Branchless Banking)

[Halaman ini sengaja dikosongkan]

BAB III
TINJAUAN PUSTAKA

3.1. Database Server

Database server adalah program komputer yang menyediakan layanan data lainnya ke komputer atau program komputer, seperti yang ditetapkan oleh model klien-server. Istilah ini juga merujuk kepada sebuah komputer yang didedikasikan untuk menjalankan program server database. Database sistem manajemen database yang sering menyediakan fungsi server, dan beberapa DBMSs (misalnya, MySQL) secara eksklusif bergantung pada model klien-server untuk akses data.

Database server menyediakan beberapa manfaat yaitu:

1. Semua data untuk organisasi dapat disimpan di satu lokasi.
2. Database server menambahkan tingkat keamanan data.
3. Database server menyediakan layanan database management service dimana data disusun
4. Dengan cara tertentu sehingga meningkatkan pencarian dan pengambilan data.
5. Beberapa client dapat mengakses data yang disimpan di database server dalam satu waktu tanpa saling mengganggu satu sama lain.

Client-server model dapat diartikan sebagai model dari suatu sistem yang membagi proses sistem antara server yang mengolah database dan client yang menjalankan aplikasi. Database server mengurangi beban akses data oleh client pada server. Database dapat diakses oleh beberapa client secara bersamaan dimana data yang diakses hanya atau diubah berasal dari satu sumber yaitu database pada server. Server tersebut diakses baik melalui suatu "front end" yang berjalan di komputer pengguna yang menampilkan data yang diminta atau "back end" yang berjalan pada server dan menangani tugas-tugas seperti analisis data dan penyimpanan. Dalam model master-slave, database server master adalah lokasi

pusat dan utama data sementara database server budak disinkronisasi backup dari master bertindak sebagai proxy. Beberapa contoh dari server basis data Oracle, DB2, Informix, Ingres, SQL Server. Setiap server menggunakan query sendiri logika dan struktur. Bahasa query SQL kurang lebih sama di semua server database.

3.2. Relational / SQL Database

Database relasional merupakan jenis Database Management System (DBMS) yang terbaru, yang memberikan gambaran atau bagan skema yang menjelaskan tentang hubungan antar tabel bisa dilakukan di dalam sebuah database. Model database ini digagas oleh seorang pakar database bernama EF codd.

Ilmu yang mempelajari tentang konsep Database Relasional disebut Database Relational System. Database relasional System merupakan konsep yang muncul setelah adanya konsep database pendahulunya yaitu network database dan hierarchical database. Dalam jenis database relasional ini, ada penggambaran yang jelas tentang hubungan suatu tabel dengan tabel yang lain bisa dilakukan, hubungan ini digambarkan dengan garis solid yang menghubungkan antara satu field name di tabel yang satu, dengan satu field name di tabel yang lain. Misalnya field name kdpasien di tabel pasien dengan field name kdpasien di tabel diagnosa pasien, yang saling terhubung karena adanya kesamaan dalam fungsi dan entitas dari objek yang dimaksud. Dengan demikian, sebuah database relasional ini dirancang untuk memiliki keterkaitan antar tabelnya, menyesuaikan dengan program atau analisa sistem yang dirancang. Dalam kerjanya struktur dari RDBMS itu tertata dan untuk *scaling* kedepannya hanya bisa *Vertical Scaling*, tapi pada dasarnya bisa juga melakukan *Horizontal Scaling* namun kurang efisien.

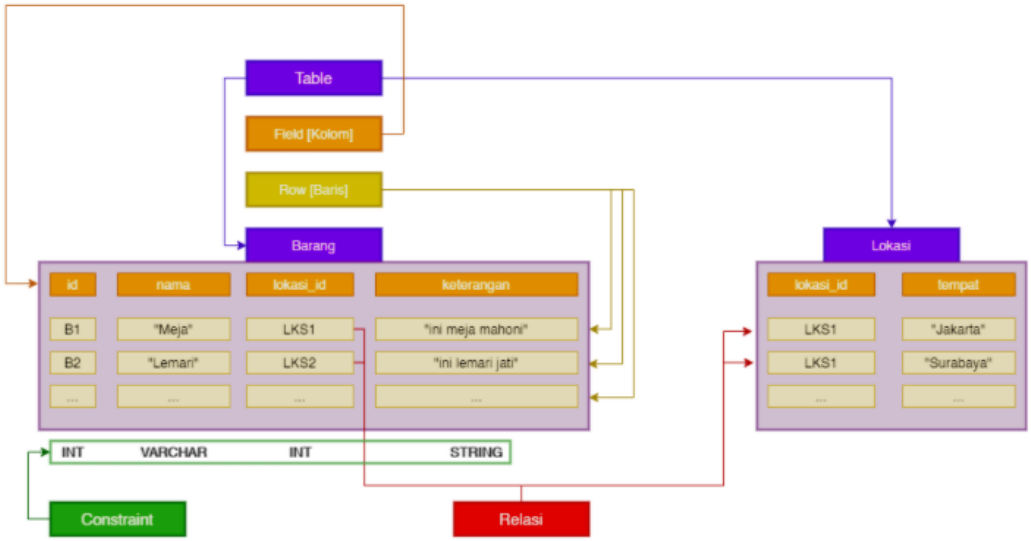

Fig. 3.2.1 Struktur Relational Database

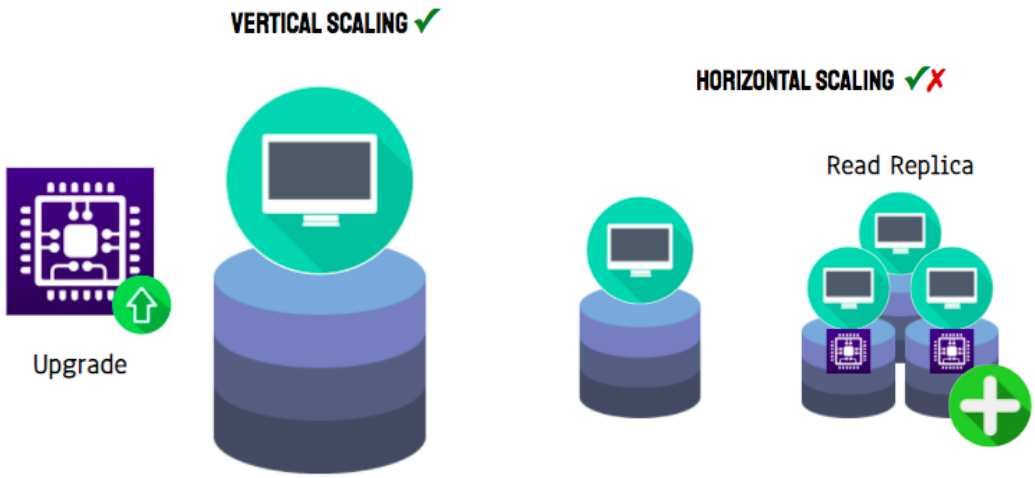

Fig. 3.2.2 Scaling dari Relational Database

3.3. Database Testing Tools

Basis data adalah bagian penting dari sistem atau aplikasi apa pun. Semuanya ada berdasarkan data. Jika tidak ada data, tidak ada yang mungkin. Ketidakstabilan database dapat menyebabkan sistem berperilaku tidak terduga. Entah database mengalami crash, atau data disimpan dalam database dengan cara yang tidak terorganisir, dalam kedua kasus data akan menjadi tidak berguna. Jadi

pengujian database membantu kita menemukan kerentanan seperti itu dalam sistem database untuk melindungi database dari keadaan tidak stabil.

Pengujian DB membutuhkan pendekatan yang terorganisir dan terencana dengan baik. Jika tim penguji tidak memiliki pemahaman yang cukup, pengujian menjadi panjang dan tidak lengkap dalam hal cakupan pengujian dan data pengujian digunakan. Tim penguji yang kurang memenuhi syarat melakukan pengujian back-box. Mereka menguji aplikasi yang bergantung pada database dengan bertindak seperti pengguna nyata, mereka hanya menguji melalui antarmuka pengguna (UI). Pengujian white-box memberikan ide tentang cakupan pengujian, tetapi membutuhkan penguji dengan keterampilan SQL yang sangat baik. Penguji tersebut lebih mahal, namun biayanya tetap terjaga, karena kueri SQL juga berfungsi untuk menguji pemicu database, prosedur, dan properti database penting lainnya. Tim penguji yang tidak berpengalaman lalai memprioritaskan fitur DB dalam hal signifikansi, yang penting untuk menghemat tenaga dan waktu. Mereka mencoba untuk menguji semuanya, kemudian kehilangan waktu dengan menguji bagian-bagian kecil dari database, dan membiarkan bagian-bagian penting tersebut tidak teruji atau tidak diuji sama sekali.

Pendekatan pengujian yang paling umum adalah mengisi database dengan jumlah data palsu yang terbatas. Data ini tidak ada hubungannya dengan data sebenarnya, sehingga tim penguji tidak dapat mengenali beberapa kekurangan.

Ketika akhirnya menemukan masalah ini, perlu menginvestasikan waktu dan upaya tambahan untuk debugging dan pengujian ulang. Data uji realistis dapat digunakan tanpa memaparkan data pelanggan yang sebenarnya. Meskipun tidak dapat menggunakan nama orang dan perusahaan sungguhan, Anda dapat menggunakan kodepos, Nomor telepon khusus untuk negara target. Dan itu dapat membantu untuk mengatasi inkonsistensi front-end yang mungkin dialami pengguna secara memadai.

Masalahnya adalah tim penguji tidak mengetahui biaya alat. Mereka cenderung menggunakan alat open source, berharap mendapatkan alat performa secara gratis. Alat ini mungkin menyediakan fungsionalitas yang kaya secara gratis, tetapi alat sumber terbuka mungkin juga kekurangan stabilitas dan kenyamanan alat komersial.Untuk layanan yang lebih baik, Anda harus membayar, jadi jika memutuskan untuk menggunakan alat komersial, penting untuk menghitung biaya dengan benar. Untuk melakukan itu perlu pemahaman yang baik tentang spesifikasi proyek dan fungsionalitas alat yang dibutuhkan.

3.4. NoSQL Database

Ketika orang menggunakan istilah "database NoSQL", mereka biasanya menggunakannya untuk merujuk ke database non-relasional. Beberapa orang mengatakan istilah "NoSQL" adalah singkatan dari "non SQL" sementara yang lain mengatakan itu adalah singkatan dari "tidak hanya SQL." Bagaimanapun, sebagian besar setuju bahwa database NoSQL adalah database yang menyimpan data dalam format selain tabel relasional.

Kesalahpahaman yang umum adalah bahwa database NoSQL atau database non-relasional tidak menyimpan data hubungan dengan baik. Database NoSQL dapat menyimpan data hubungan — mereka hanya menyimpannya secara berbeda dari database relasional. Faktanya, jika dibandingkan dengan database SQL, banyak yang menganggap data hubungan pemodelan di database NoSQL lebih mudah daripada di database SQL, karena data terkait tidak harus dipisahkan antar tabel. Model data NoSQL memungkinkan data terkait disarankan dalam satu struktur data.

Database NoSQL muncul pada akhir tahun 2000-an karena biaya penyimpanan menurun drastis. Lewatlah sudah hari-hari kebutuhan untuk membuat model data yang kompleks dan sulit dikelola hanya untuk tujuan mengurangi duplikasi data. Pengembang (bukan penyimpanan) menjadi biaya utama

pengembangan perangkat lunak, sehingga database NoSQL dioptimalkan untuk produktivitas pengembang.

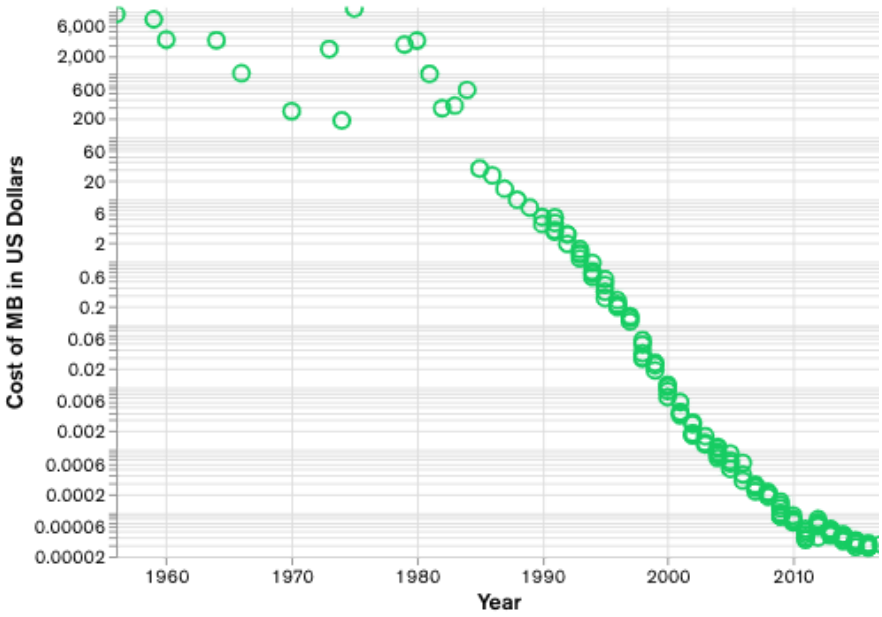

Fig. 3.4.1 Biaya Penyimpanan menurun tiap tahun [8]

Karena biaya penyimpanan menurun dengan cepat, jumlah aplikasi data yang diperlukan untuk menyimpan dan kueri meningkat. Data ini datang dalam berbagai bentuk dan ukuran — terstruktur, semistruktur, dan polimorfik — dan mendefinisikan skema sebelumnya hampir mustahil. Basis data NoSQL memungkinkan pengembang untuk menyimpan data tidak terstruktur dalam jumlah besar, memberi mereka banyak fleksibilitas.

Selain itu, Agile Manifesto semakin populer, dan insinyur perangkat lunak memikirkan kembali cara mereka mengembangkan perangkat lunak. Mereka menyadari kebutuhan untuk cepat beradaptasi dengan kebutuhan yang berubah. Mereka membutuhkan kemampuan untuk melakukan iterasi dengan cepat dan membuat perubahan di seluruh tumpukan perangkat lunak mereka — hingga ke model database. Database NoSQL memberi mereka fleksibilitas ini.

Komputasi awan juga semakin populer, dan pengembang mulai menggunakan awan publik untuk menghosting aplikasi dan data mereka. Mereka menginginkan kemampuan untuk mendistribusikan data ke beberapa server dan wilayah untuk membuat aplikasi mereka tangguh, untuk melakukan penskalaan alih-alih penskalaan, dan untuk menempatkan data secara cerdas secara geografis. Beberapa database NoSQL seperti MongoDB menyediakan kemampuan ini.

3.5. Teorema CAP

Di masa lalu, ketika kami ingin menyimpan lebih banyak data atau meningkatkan kekuatan pemrosesan kami, opsi yang umum adalah menskalakan secara vertikal (mendapatkan mesin yang lebih kuat) atau lebih mengoptimalkan basis kode yang ada. Namun, dengan kemajuan dalam pemrosesan paralel dan sistem terdistribusi, lebih umum untuk memperluas secara horizontal, atau memiliki lebih banyak mesin untuk melakukan tugas yang sama secara paralel. Kita sudah bisa melihat banyak alat manipulasi data di proyek Apache seperti Spark, Hadoop, Kafka, Zookeeper, dan Storm. Namun, untuk secara efektif memilih alat pilihan, ide dasar Teorema CAP diperlukan. Teorema CAP adalah sebuah konsep bahwa sistem database terdistribusi hanya dapat memiliki 2 dari 3: **Konsistensi, Ketersediaan, dan Toleransi Partisi.**

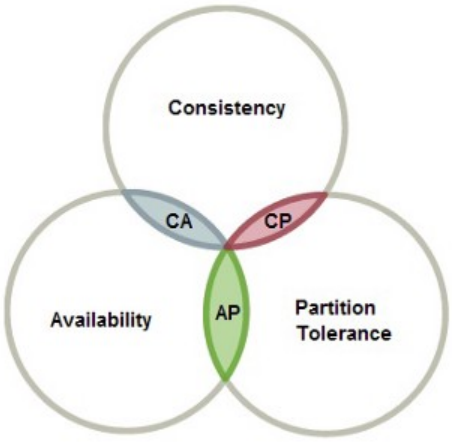

Fig. 3.5.1 Visualisasi Graf Teorema CAP

Teorema CAP sangat penting dalam dunia Big Data, terutama ketika kita perlu melakukan trade off di antara ketiganya, berdasarkan kasus penggunaan unik kami. Di blog ini, saya akan mencoba menjelaskan masing-masing konsep ini dan alasan trade off. Saya akan menghindari penggunaan contoh spesifik karena DBMS berkembang pesat.

1. Toleransi Partisi

Kondisi ini menyatakan bahwa sistem tetap berjalan meskipun sejumlah pesan mengalami penundaan oleh jaringan antar node. Sistem yang toleran terhadap partisi dapat mempertahankan sejumlah kegagalan jaringan yang tidak mengakibatkan kegagalan seluruh jaringan. Catatan data direplikasi secara memadai di seluruh kombinasi node dan jaringan untuk menjaga sistem tetap aktif melalui pemadaman yang terputus-putus. Saat berhadapan dengan sistem terdistribusi modern, Partition Tolerance bukanlah pilihan. Itu suatu kebutuhan. Karenanya, kita harus berdagang antara Konsistensi dan Ketersediaan.

2. Konsistensi

Kondisi ini menyatakan bahwa semua node melihat data yang sama pada waktu yang bersamaan. Sederhananya, melakukan operasi baca akan mengembalikan nilai operasi tulis terbaru yang menyebabkan semua node mengembalikan data yang sama. Suatu sistem memiliki konsistensi jika transaksi dimulai dengan sistem dalam keadaan konsisten, dan diakhiri dengan sistem dalam keadaan konsisten. Dalam model ini, sistem dapat (dan memang) bergeser ke keadaan tidak konsisten selama transaksi, tetapi seluruh transaksi akan dibatalkan jika ada kesalahan selama tahap mana pun dalam proses tersebut. Dalam gambar, kami memiliki 2 rekaman berbeda ("Bulbasaur" dan "Pikachu") di stempel waktu yang berbeda. Output pada partisi ketiga adalah “Pikachu”, input terbaru. Namun, node memerlukan waktu untuk memperbarui dan tidak akan sering tersedia di jaringan.

3. Ketersediaan

Kondisi ini menyatakan bahwa setiap request mendapat respon sukses / gagal. Mencapai ketersediaan dalam sistem terdistribusi mengharuskan sistem tetap beroperasi 100% setiap saat. Setiap klien mendapat respons, terlepas dari status node individual mana pun dalam sistem. Metrik ini mudah untuk diukur: Anda bisa mengirimkan perintah baca / tulis, atau Anda tidak bisa. Oleh karena itu, database tidak bergantung waktu karena node harus tersedia secara online setiap saat. Artinya, tidak seperti contoh sebelumnya, kita tidak tahu apakah "Pikachu" atau "Bulbasaur" ditambahkan terlebih dahulu. Outputnya bisa salah satu. Karenanya, ketersediaan tinggi tidak dapat dilakukan saat menganalisis data streaming pada frekuensi tinggi.

3.6. MySQL

Fig. 3.6.1 MySQL Logo [7]

MySQL merupakan sebuah perangkat lunak sistem manajemen basis data SQL (bahasa inggris : data management system) atau DBMS yang multithread, multi-user, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL . Tidak seperti Apache yang merupakan software yang dikembangkan oleh komunitas umum, dan cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia yaitu MySQL AB. MySQL AB memegang penuh hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah : david axmark, allan larsson, dan Michael "monty widenius.

Kelebihan MySQL antara lain :

1. Portabilitas. MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lagi.
2. Free (bebas didownload) MySQL didistribusikan secara open source, dibawah lisensi GPL sehingga dapat digunakan secara cuma-cuma.
3. stabil dan tangguh, fleksibel dengan berbagai pemrograman
4. Security yang baik & mendukung transaksi
5. dukungan dari banyak komunitas & perkembangan software yang cukup cepat
6. kemudahan management database

3.7. PostgreSQL

Fig. 3.7.1 PostgreSQL Logo [7]

PostgreSQL hadir dengan banyak fitur yang bertujuan untuk membantu pengembang membangun aplikasi, administrator untuk melindungi integritas data dan membangun lingkungan yang toleran terhadap kesalahan, dan membantu Anda mengelola data Anda tidak peduli seberapa besar atau kecil kumpulan data tersebut. Selain gratis dan open source, PostgreSQL sangat dapat dikembangkan. Misalnya, Anda dapat menentukan tipe data Anda sendiri, membuat fungsi kustom, bahkan menulis kode dari bahasa pemrograman yang berbeda tanpa mengkompilasi ulang database Anda!

PostgreSQL mencoba menyesuaikan dengan standar SQL di mana kesesuaian tersebut tidak bertentangan dengan fitur tradisional atau dapat menyebabkan keputusan arsitektur yang buruk. Banyak fitur yang dibutuhkan oleh standar SQL didukung, meskipun terkadang dengan sintaks atau fungsi yang sedikit

berbeda. Pergerakan lebih lanjut menuju kesesuaian dapat diharapkan dari waktu ke waktu. Pada rilis versi 13 pada September 2020, PostgreSQL memenuhi setidaknya 170 dari 179 fitur wajib untuk kesesuaian SQL: 2016 Core. Sampai tulisan ini dibuat, tidak ada database relasional yang memenuhi kesesuaian penuh dengan standar ini.

Kelebihan PostgreSQL

- Terpercaya dan Stabil
- Merupakan Database objek-relasional
- Menangani konkurensi lebih baik
- mengimplementasikan Multiversion Concurrency Control (MVCC) tanpa kunci baca

3.8. MongoDB

Fig.3.8.1 MongoDB Logo [7]

MongoDB adalah database NoSQL berorientasi dokumen yang digunakan untuk penyimpanan data volume tinggi. Alih-alih menggunakan tabel dan baris seperti pada database relasional tradisional, MongoDB menggunakan koleksi dan dokumen. Dokumen terdiri dari pasangan nilai kunci yang merupakan unit dasar data di MongoDB. Koleksi berisi sekumpulan dokumen dan fungsi yang setara dengan tabel database relasional. MongoDB adalah database yang muncul sekitar pertengahan 2000-an. Di bawah ini adalah beberapa alasan mengapa seseorang harus mulai menggunakan MongoDB

Berorientasi pada dokumen - Karena MongoDB adalah database tipe NoSQL, alih-alih memiliki data dalam format tipe relasional, ia menyimpan data

dalam dokumen. Ini membuat MongoDB sangat fleksibel dan mudah beradaptasi dengan situasi dan persyaratan dunia bisnis nyata. Kueri ad hoc - MongoDB mendukung pencarian berdasarkan field, query range, dan pencarian ekspresi reguler. Pertanyaan dapat dibuat untuk mengembalikan bidang tertentu dalam dokumen. Pengindeksan - Indeks dapat dibuat untuk meningkatkan kinerja pencarian dalam MongoDB. Semua bidang dalam dokumen MongoDB dapat diindeks. Replikasi - MongoDB dapat menyediakan ketersediaan tinggi dengan set replika. Satu set replika terdiri dari dua atau lebih instans DB mongo. Setiap anggota kumpulan replika dapat bertindak dalam peran replika utama atau sekunder kapan saja. Replika utama adalah server utama yang berinteraksi dengan klien dan melakukan semua operasi baca / tulis. Replika sekunder menyimpan salinan data utama menggunakan replikasi bawaan. Ketika replika utama gagal, kumpulan replika secara otomatis beralih ke replika sekunder dan kemudian menjadi server utama. Load balancing - MongoDB menggunakan konsep sharding untuk menskalakan secara horizontal dengan membagi data di beberapa instance MongoDB. MongoDB dapat berjalan di beberapa server, menyeimbangkan beban dan / atau menggandakan data untuk menjaga sistem tetap aktif dan berjalan jika terjadi kegagalan perangkat keras.

MongoDB diketahui digunakan oleh Barclays, Bosch, Cisco, Pfizer, Kota Chicago, Codecademy, Coinbase, eBay, Foursquare, HSBC, IBM, Orange S.A ., Sega, The Gap, Inc ., Uber, Urban Outfitters, dan Imigrasi AS dan Penegakan Bea Cukai.

3.9. Cassandra

Fig. 3.9.1 Apache Cassandra Logo [7]

Apache Cassandra adalah database NoSQL open source terdistribusi yang dimulai secara internal di Facebook dan dirilis sebagai proyek open-source pada Juli 2008. Cassandra memberikan ketersediaan berkelanjutan (zero downtime), kinerja tinggi, dan skalabilitas linier yang dibutuhkan aplikasi modern, sementara juga menawarkan kesederhanaan operasional dan replikasi yang mudah di seluruh pusat data dan geografi. Cassandra dapat menangani informasi berukuran petabyte dan ribuan operasi bersamaan per detik, memungkinkan organisasi untuk mengelola data dalam jumlah besar di lingkungan cloud hybrid dan multi cloud.

Di DataStax, kami bekerja keras dengan komunitas sumber terbuka untuk membangun kedewasaan Cassandra selama lebih dari satu dekade guna memperkuat posisinya sebagai database terkemuka untuk aplikasi cloud-native.

Apache Cassandra dikembangkan oleh Avinash Lakshman dan Prashant Malik saat keduanya bekerja sebagai insinyur di Facebook. Basis data dirancang untuk mendukung fitur pencarian kotak masuk Facebook, sehingga memudahkan pengguna untuk menemukan percakapan dan konten lain yang mereka cari dengan cepat. Arsitektur menggabungkan model distribusi yang diusulkan dalam makalah Amazon Dynamo untuk memungkinkan penskalaan horizontal di beberapa node dengan mesin penyimpanan terstruktur log yang dijelaskan dalam makalah

BigTable Google. Hasilnya adalah database yang sangat skalabel yang dapat menangani kasus penggunaan yang paling kaya data dan performa intensif.

Pada Juli 2008, Facebook membuka situs Cassandra. Pada Maret 2009, Cassandra menjadi proyek Apache Incubator. Pada April 2010, ia lulus dari inkubator, menjadi proyek tingkat atas untuk Apache Foundation. Saat ini, Cassandra tersedia secara gratis di bawah Lisensi Apache 2.0. Tim di DataStax adalah pemimpin dalam evolusi basis data sumber terbuka, bertanggung jawab atas sebagian besar komitmen kode proyek melalui rilis 3.0, dan kami telah mendedikasikan kembali diri kami sebagai kolaborator aktif untuk masa depan proyek, membantu dengan rilis 4.0 dan seterusnya.

Open Source

Organisasi pengembangan perangkat lunak modern telah banyak bergerak untuk mengadopsi teknologi open source, dimulai dengan sistem operasi Linux dan berkembang menjadi infrastruktur untuk mengelola data. Teknologi open source menarik karena keterjangkauan dan ekstensibilitasnya, serta fleksibilitas untuk menghindari penguncian vendor. Organisasi yang mengadopsi laporan open source kecepatan inovasi yang lebih tinggi dan adopsi yang lebih cepat.

Antarmuka yang fleksibel dan familier

Antarmuka yang Fleksibel dan Akrab: Cassandra Query Language (CQL) mirip dengan SQL, yang berarti sebagian besar pengembang harus memiliki waktu yang cukup mudah untuk memahaminya. (Berikut adalah pengantar CQL jika Anda membutuhkan bantuan).

Performa tinggi

Kinerja Tinggi: Mayoritas database tradisional menampilkan arsitektur primer / sekunder. Dalam konfigurasi ini, replika primer tunggal menjalankan operasi baca dan tulis, sedangkan replika sekunder hanya dapat melakukan operasi baca. Kerugian arsitektur ini termasuk peningkatan latensi, serta biaya yang lebih

tinggi dan ketersediaan yang lebih rendah dalam skala besar. Di Cassandra, tidak ada satu node pun yang bertugas mereplikasi data di seluruh cluster. Sebaliknya, setiap node mampu melakukan semua operasi baca dan tulis. Ini meningkatkan kinerja dan menambah ketahanan ke database.

3.10. Redis

Fig. 3.10.1 Redis Logo [7]

Redis adalah sumber terbuka (berlisensi BSD), penyimpanan struktur data dalam memori, digunakan sebagai database, cache, dan perantara pesan. Redis menyediakan struktur data seperti string, hash, daftar, set, set yang diurutkan dengan kueri rentang, bitmap, hyperloglog, indeks geospasial, dan aliran. Redis memiliki replikasi bawaan, pembuatan skrip Lua, pengusiran LRU, transaksi, dan berbagai tingkat persistensi pada disk, dan menyediakan ketersediaan tinggi melalui Redis Sentinel dan partisi otomatis dengan Redis Cluster.

Anda dapat menjalankan operasi atom pada jenis ini, seperti menambahkan ke string; menaikkan nilai dalam hash; mendorong elemen ke daftar; komputasi himpunan persimpangan, persatuan dan perbedaan; atau mendapatkan anggota dengan peringkat tertinggi dalam kumpulan yang diurutkan.

Untuk mencapai kinerja terbaik, Redis bekerja dengan kumpulan data dalam memori. Bergantung pada kasus penggunaan Anda, Anda dapat mempertahankan data baik dengan membuang kumpulan data secara berkala ke disk atau dengan menambahkan setiap perintah ke log berbasis disk. Anda juga dapat menonaktifkan

persistensi jika Anda hanya memerlukan cache dalam memori yang kaya fitur dan berjaringan.

Redis juga mendukung replikasi asinkron, dengan sinkronisasi pertama non-pemblokiran yang sangat cepat, koneksi ulang otomatis dengan sinkronisasi ulang sebagian pada pembagian bersih.Berikut ini beberapa keunggulan Redis. Sangat cepat - Redis sangat cepat dan bisa melakukan sekitar 110000 SET per detik, sekitar 81000 GET per detik. Mendukung tipe data yang kaya - Redis secara native mendukung sebagian besar tipe data yang telah diketahui pengembang seperti list, set, sorted, dan hash yang diurutkan. Hal ini memudahkan pemecahan berbagai masalah sebagai mana kita tahu masalah yang bisa ditangani dengan lebih baik dengan tipe data. Operasi bersifat atom - Semua operasi Redis bersifat atom, yang memastikan bahwa jika dua klien mengakses secara bersamaan, server Redis akan menerima nilai yang diperbarui. Multi-utility tool - Redis memiliki multi-utility tool dan dapat digunakan dalam sejumlah kasus penggunaan seperti caching, messaging-queues (Redis native mendukung Publish / Subscribe), data singkat dalam aplikasi Anda, seperti web *sessions* aplikasi, *web page hit counts*, dll.

3.11. Jmeter

Fig. 3.11.1 Apache Jmeter Logo [2]

Apache JMeter adalah perangkat lunak open source Java murni, yang pertama kali dikembangkan oleh Stefano Mazzocchi dari Apache Software Foundation, dirancang untuk membuat uji perilaku fungsional dan mengukur kinerja. Anda dapat menggunakan JMeter untuk menganalisis dan mengukur kinerja aplikasi web atau berbagai layanan. Pengujian Kinerja berarti menguji aplikasi web terhadap beban berat, banyak dan lalu lintas pengguna secara

bersamaan. JMeter awalnya digunakan untuk menguji Aplikasi Web atau aplikasi FTP. Saat ini, digunakan untuk uji fungsional, uji server database, dll.

Pernahkah Anda menguji server web untuk mengetahui seberapa efisien kerjanya? Berapa banyak pengguna bersamaan yang dapat ditangani server web? Katakanlah suatu hari, atasan Anda meminta Anda melakukan pengujian kinerja www.google.com untuk 100 pengguna. Apa yang akan kamu lakukan?

Tidaklah mungkin mengatur 100 orang dengan PC dan akses internet secara bersamaan mengakses google.com. Pikirkan persyaratan infrastruktur saat Anda menguji 10.000 pengguna (jumlah kecil untuk situs seperti google). Oleh karena itu Anda memerlukan alat perangkat lunak seperti JMeter yang akan mensimulasikan perilaku pengguna nyata dan kinerja / uji beban situs Anda.

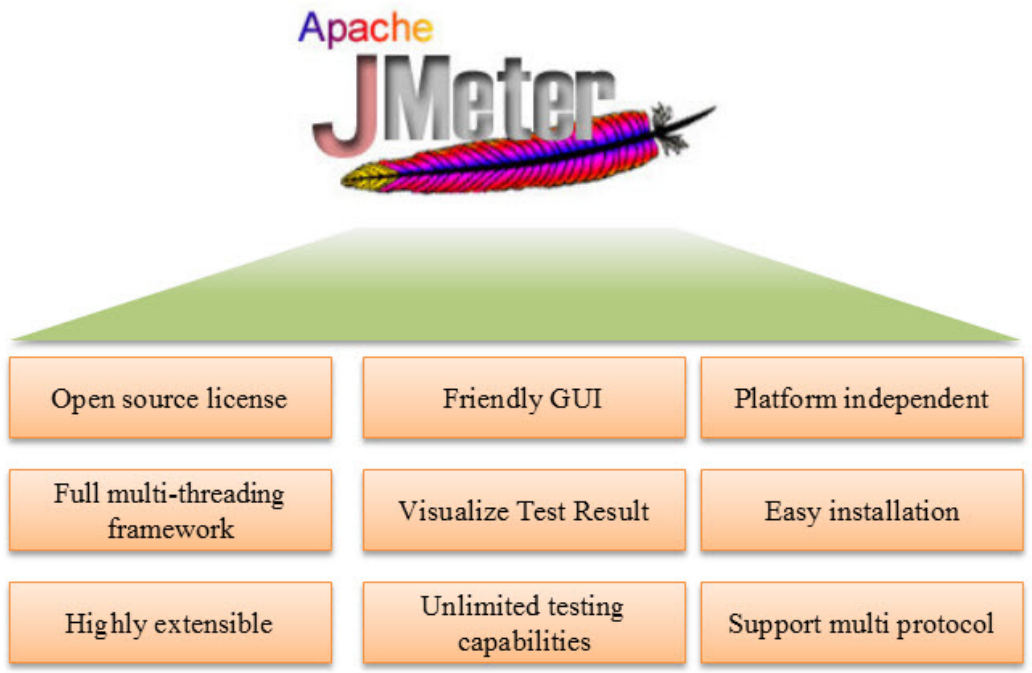

Fig. 3.11.2 Kelebihan Apache Jmeter [2]

Alur kerja dasar JMeter seperti yang ditunjukkan pada gambar di bawah ini

JMeter mensimulasikan sekelompok pengguna yang mengirimkan permintaan ke server target, dan mengembalikan informasi statistik dari server target melalui diagram grafis

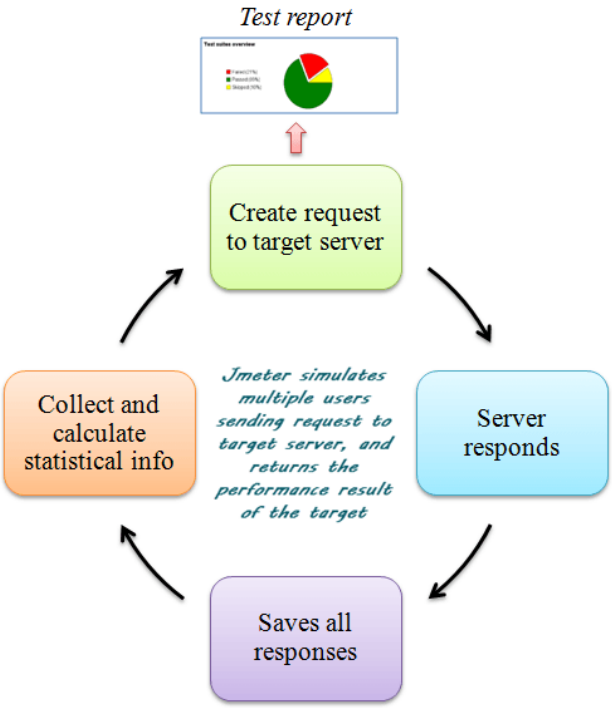

Fig. 3.11.3 Penggunaan Jmeter [2]

3.12. SoapUI

Fig. 3.12.1 SoapUI Logo [7]

SoapUI adalah alat untuk menguji Layanan Web; ini bisa berupa SOAP Web Services serta RESTful Web Services atau HTTP based services. SoapUI adalah Open Source dan alat yang sepenuhnya gratis dengan pendamping komersial -SoapUI Pro- yang memiliki fungsionalitas ekstra untuk perusahaan dengan

Layanan Web penting.

SoapUI telah diunduh lebih dari 3 juta kali dan dipandang sebagai standar de facto untuk Pengujian Layanan API. Ini berarti ada banyak pengetahuan tentang alat ini di luar sana, baca blog di internet untuk info lebih lanjut tentang penggunaan SoapUI di kehidupan nyata. Kami menghargai setiap unduhan dan bekerja sangat keras untuk menciptakan produk super untuk Anda.

SoapUI dapat digunakan untuk menyelesaikan pengujian RESTful API dan SOAP Web Service. Anda dapat melakukan Pengujian Fungsional, Pengujian Performa, Pengujian Interoperabilitas, Pengujian Regresi, dan banyak lagi. Kami bertujuan agar pengujian mudah dilakukan, misalnya untuk membuat Pengujian Beban, Anda cukup mengklik kanan pengujian fungsional dan menjalankannya sebagai pengujian beban. Anda dapat mensimulasikan Layanan Web. Anda dapat merekam tes dan menggunakannya Nanti. Anda dapat membuat potongan kode dari WSDL. Anda bahkan dapat membuat spesifikasi REST (WADL) dari komunikasi yang direkam. Ada banyak hal yang dapat Anda lakukan, kami mendorong Anda untuk melihat-lihat dokumentasi dan bermain-main dengan alat tersebut.

[Halaman ini sengaja dikosongkan]

BAB IV
HASIL TINJAUAN LITERATUR

4.1. Pemilihan Sistem

Data adalah hal penting dalam era modern. Data dapat dimanfaatkan sedemikian rupa sehingga dapat digunakan untuk keperluan tertentu. Data yang dikumpulkan dapat diolah sehingga memunculkan pola yang dapat dianalisis lebih lanjut untuk memprediksi kejadian tertentu. Penggunaan data saat ini sangat tinggi sehingga setiap saat akan dijumpai lalu lintas data kapanpun dan dimanapun. Peradaban yang semakin maju memungkinkan pertukaran data yang sangat dinamis dan fleksibel.

Telah banyak aplikasi manajemen database dengan paradigma terkhusus database big data yaitu NoSQL. NoSQL sendiri memiliki berbagai sistem yang juga memiliki kegunaan. Masing-masing memiliki keuntungannya sendiri dari segi kinerja maupun metode yang diberikan. Sehingga, pengembang dapat memilih sesuai kebutuhan aplikasi yang dikembangkan. Dengan penggunaan CAP Theorem dapat didapat suatu kesimpulan database mana dari SQL dan NoSQL yang dipakai, darisitu saya menemukan titik temunya yaitu dari:

```
* small project + low scale + unknown access pattern -> RDBMS
* large project + high scale + relational query -> RDBMS + Read Replicas
* medium/large project + high scale + high performance -> NOSQL
```

Fig 4.1.1 Visualisasi Penentuan Database

Dari situ tertera jelas bahwasanya untuk proyek sebesar jalan tol yang membutuhkan banyak transaksi yang nonstop maka dipilihlah NoSQL. Dari NoSQL pun kami pilih beberapa sistem manajemen basis data yang populer adalah MongoDB, Cassandra, dan Redis. Sistem tersebut telah banyak ditemui di perusahaan besar seperti facebook, google dan amazon. Untuk hasil data yang lebih

umum, saya menggunakan 3 sistem yang digunakan dengan fungsi berbeda satu sama lain.

Salah satu proses penting dalam sistem manajemen basis data adalah replikasi, yaitu kemampuan sistem untuk melakukan penggandaan pada suatu state tertentu yang digunakan sebagai data backup maupun validasi terhadap data utama. Proses ini dapat dijumpai hampir pada seluruh sistem manajemen basis data. Secara umum, replikasi digunakan untuk mencegah hilangnya data atau korupsi data yang mungkin terjadi akibat kesalahan pemrosesan pada sistem. selain replikasi, arsitektur serta studi kasus nyata digunakan untuk memperkokoh studi literatur ini.

4.2. Metode Replikasi

Setiap sistem manajemen basis data memiliki metode replikasi masing-masing. Pada dasarnya proses replikasi pada basis data adalah melakukan penyalinan secara berkala pada basis data suatu mesin ke mesin lainnya sehingga setiap mesin memiliki tingkat informasi yang sama. Hasilnya adalah sebuah basis data yang terdistribusi dimana pengguna dapat mengakses data yang relevan tanpa mengganggu pengaksesan pada pengguna lainnya. Replikasi pada basis data dapat terjadi sekali atau sebagai proses yang berjalan secara terus-menerus. Secara umum, sistem manajemen basis data terdistribusi bekerja untuk memastikan perubahan, penambahan, dan penghapusan yang dijalankan terhadap data pada suatu lokasi direfleksikan kepada seluruh lokasi yang berhubungan terhadap data tersebut.

1. MongoDB

Replikasi pada MongoDB dilakukan menggunakan replica set yaitu suatu kelompok proses server MongoDB yang menjaga himpunan data yang sama. Replica set memberikan kebebasan terhadap redundancy dan high availability. Seluruh operasi tersebut dilakukan secara asinkron. Replica Set akan memiliki beberapa node yang menyimpan data dan satu arbiter node yang bersifat opsional. Pada node yang menyimpan data, salah satu dan hanya satu node yang akan menjadi primary node, sisanya akan bekerja sebagai secondary node. Primary node

akan menerima seluruh operasi write. Primary node akan menyimpan semua perubahan pada data yang disimpannya kepada riwayat operasinya.

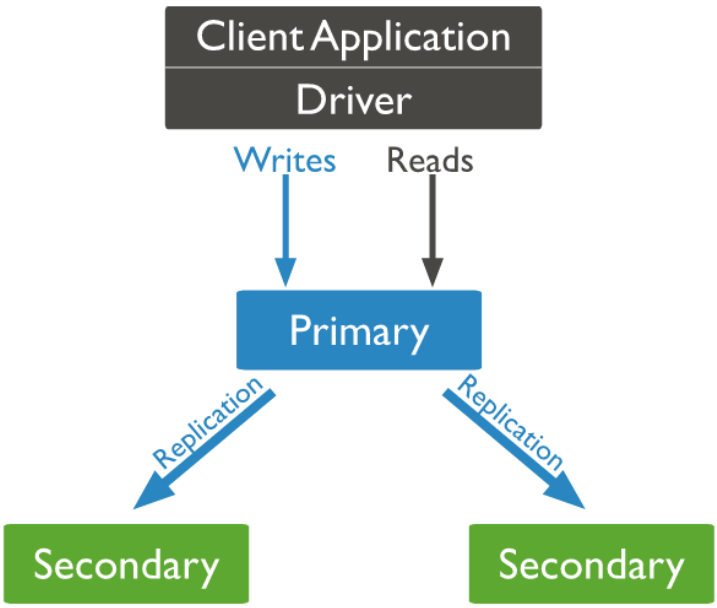

Fig 4.2.1.1 Arsitektur MongoDB[12]

Secondary node akan menggandakan setiap riwayat operasi primary node sehingga node tersebut akan memiliki data yang sama seperti pada primary node. Saat primary node tidak tersedia, secondary node akan melakukan pemilihan untuk menjadikan salah satu secondary node sebagai primary node. Pada beberapa situasi, replica set dapat membuat suatu arbiter node sebagai pengganti secondary node. Arbited node hanya akan berfungsi saat pemilihan primary node muncul dan tidak menyimpan state apapun. Arbiter node akan membantu dalam pemilihan dan state-nya tidak akan berubah.

Pada saat primary node tidak melakukan komunikasi kepada anggota node lainnya pada suatu periode tertentu, secondary node yang memenuhi persyaratan akan melakukan pemilihan dan mencalonkan dirinya sendiri sebagai primary node yang baru. Setelah pemilihan himpunan node akan melakukan operasi seperti biasa dengan primary node yang baru. Saat proses pemilihan, replica set tidak dapat melakukan operasi write dan harus menunggu hingga proses pemilihan selesai.

MongoDB menyediakan konfigurasi untuk melakukan operasi read pada secondary node sehingga saat proses pemilihan, pengguna dapat melakukan pembacaan data tanpa primary node. Consistency Model yang digunakan pada MongoDB adalah Causal Consistency. Jika Causal Consistency menerapkan prinsip durabilitas, maka seluruh operasi write dan read akan membutuhkan persetujuan mayoritas dari anggota replica set. Sebaliknya, jika tidak maka operasinya tidak menjamin kesamaan data. Hal ini menunjukkan bahwa protokol yang dimiliki oleh replica set tersebut berupa Replicated-Write Protocol.

2. Cassandra

Arsitektur replikasi Cassandra berupa coordinator node dan beberapa replica node. Kemudian node tersebut akan membentuk ring yang saling berhubungan dengan tetangganya. Terdapat dua strategi replikasi pada Cassandra. Strategi pertama berupa simple strategy dimana seluruh ring menjadi satu dan berkomunikasi satu sama lain, kemudian strategi selanjutnya berupa network topology strategi dengan membentuk beberapa kelompok khusus pada arsitekturnya.

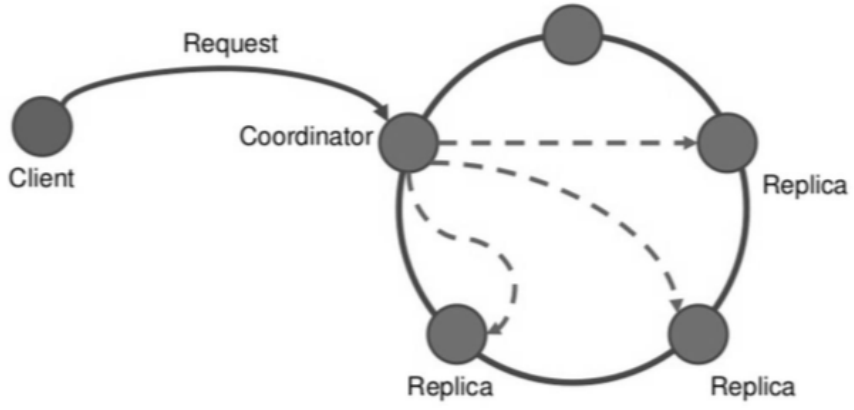

Fig. 4.2.2.1 Simple Strategy pada replikasi Cassandra [13]

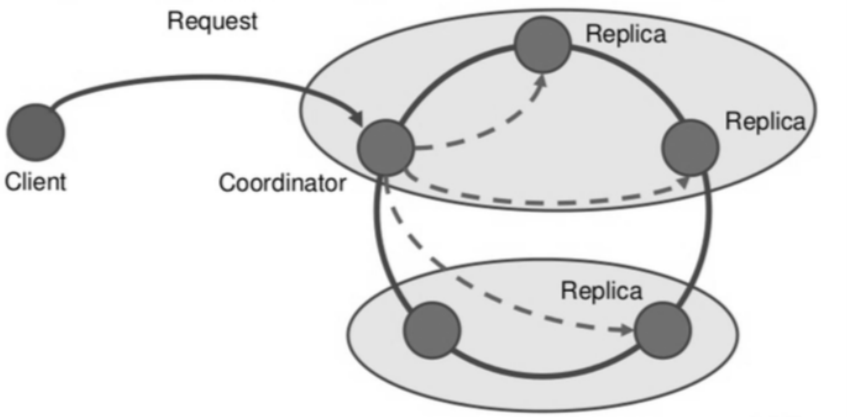

Fig. 4.2.2.2 Network Topology Strategy pada replikasi Cassandra [13]

Cassandra memiliki fitur untuk mengatur tingkat konsistensi nya, Secara umum consistency model yang digunakan berupa Eventual Consistency dengan Replicated-Write Protocol.

3. Redis

Replikasi pada Redis cukup sederhana dengan mengandalkan master-slave replication yang bekerja secara asinkron. Slave akan selalu memeriksa semua data yang diproses pada replication stream yang diberikan oleh master. Mater akan selalu menangani query saat terdapat satu atau lebih slave yang sedang melakukan sinkronisasi awal. Saat itu juga, slave dapat menangani query menggunakan dataset lama.

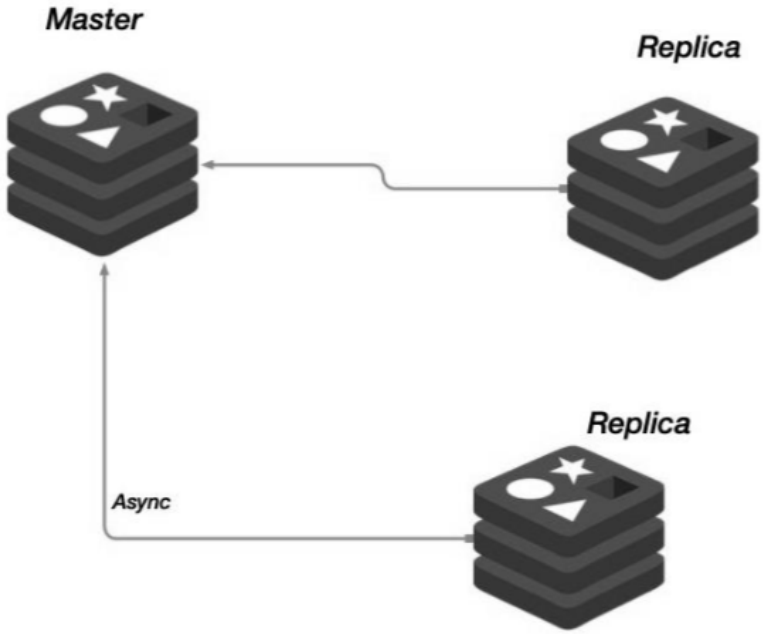

Fig. 4.2.3.1 Arsitektur Replikasi Redis [14]

Redis juga memiliki tingkat konsistensi yang dapat dikonfigurasi. Consistency model yang digunakan adalah eventual consistency dengan Primary-Based Protocol.

4.3. Studi Kasus

Pada bab studi kasus ini akan dibahas lebih detail tentang penggunaan dari ketiga basis data diatas dalam industri. Dimulai dari Apache Cassandra, salah satu studi kasus yang dapat dibahas adalah pada perusahaan Activision inc. yang bertempat di California,Amerika Serikat. Activision bertanggung jawab untuk memastikan penggemar Call of Duty di Dunia dapat mengakses aplikasi melalui akun pengguna dengan fitur yang lengkap. Oleh sebab itu, Activision harus memilih database yang datanya harus benar dan tersedia setiap saat alhasil mereka melakukan migrasi data konfigurasi global dan data secara langsung pada Apache Cassandra. Cerita mengenai Migrasi ini menurut Direktur Consumer Technology Activision, Daryl Kanhouse “Perusahaan memulai proyek untuk meningkatkan cara mereka menggunakan data untuk memberikan pengalaman pelanggan yang lebih baik pada tahun 2011. Mereka mencoba banyak database yang berbeda, termasuk Oracle, MongoDB dan Infobright, dan sering beralih ketika mencoba menemukan database yang dapat diselesaikan. semua tantangannya.Pada tahun 2014, mereka mencoba Apache Cassandra - dan untuk pertama kalinya dalam lima tahun, tidak beralih ke yang lain, kata Kanhouse. Cassandra adalah database NoSQL bersumber terbuka dan dapat diskalakan yang bertindak sebagai platform untuk aplikasi yang membutuhkan kinerja cepat dan tidak ada waktu henti. Dengan dirilisnya Call of Duty: Advanced Warfare pada tahun 2014, Activision mencoba sistem baru di mana, berdasarkan data waktu nyata, dapat mengirim pesan kepada para pemain dengan komunikasi yang sangat dipersonalisasi untuk meningkatkan pengalaman pemain dan meningkatkan keterlibatan. Ini melibatkan data dalam jumlah besar, dan menurut Kanhouse, perusahaan 'tidak dapat melakukannya tanpa Cassandra'. ”[9]

Selanjutnya untuk studi kasus MongoDB, dapat dilihat dari foursquare, salah satu aplikasi berbasis lokasi yang mulai populer sejak tahun 2009. Dalam pihak foursquare yang memiliki sumber daya teknisi yang terbatas dan pengguna serta aktivitas data dari aplikasi semakin meningkat, mereka membuat suatu keputusan untuk migrasi data dari lokasi-lokasi yang ada dan data check-in ke MongoDB untuk penggunaan dengan skala waktu yang Panjang. Pada awalnya, pihak foursquare menggunakan basis data relasional, mereka juga telah memisah database menjadi dua node yaitu satu untuk check-in yang merupakan data terbesar, dan node satu lagi untuk sisanya. Namun data check-in sendiri saja dapat berkembang melampaui kemampuan dari node tersebut, disitulah MongoDB dipakai untuk menangani masalah tersebut dan untuk pengembangan lebih lanjut. Foursquare memakai salah satu fitur MongoDB yaitu built-in auto-sharding yang berfungsi untuk membagi database yang membuat foursquare dapat mengakses dan membuat node baru dengan mudah seiring dengan berkembangnya aplikasi.[10]

Terakhir untuk studi kasus pada Redis, ada pada perusahaan Coffee Meets Bagel. Perusahaan ini bergerak dalam industri aplikasi dating. Perusahaan ini membedakan dirinya dengan aplikasi serupa menggunakan fitur authentic matching yaitu membatasi jumlah kecocokan pasangan setiap hari. Pada awalnya perusahaan ini menggunakan Cassandra sebelum pindah menggunakan Redis dengan tujuan untuk memanfaatkan Redis Enterprise VPC yang berfungsi untuk mempercepat performa dari proses caching, user analytics, dan transaksi data secara real-time. Perusahaan ini dirintis pada tahun 2012 dan ingin mengembangkan aplikasinya dalam skala global dengan pemrosesan data sebesar 1TB per jam. Untuk itu Coffee Meets Bagel memerlukan basis data yang mumpuni. Redis Enterprise menyediakan perubahan skala secara otomatis, proses failover, kemampuan untuk membuat cluster database, layanan yang persistence, dan juga bersifat high-availability. Hal itu membuat Coffee Meets Bagel dapat fokus untuk pengembangan aplikasinya.[11]

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Kesimpulan yang didapat setelah melakukan studi literatur perbandingan database yang tepat untuk kasus high traffic load transaksi jalan tol ialah:

1. Dalam menentukan sistem database yang tepat seperti halnya kasus transaksi jalan tol yang membutuhkan high traffic load, yaitu dengan poin Scaling serta Struktur. Kasus KP ini lebih cocok untuk melakukan *Vertical Scaling* yang dimana artinya dapat melakukan lebih dari satu perangkat mesin database tanpa dibatasi host yang sama serta Struktur yang fleksibel. Dua hal itu merupakan ciri-ciri NoSQL sehingga ini merupakan database yang dipilih.
2. Untuk poin database NoSQL mana yang dipilih, dilihat dari poin Replikasi data, Arsitektur, dan Studi Kasus nyata. Dari hasil riset yang didapat Apache Cassandra adalah hal yang tepat untuk kasus ini karena sistem serta replikasi data cepat nan fleksibel dan studi kasus nyata memiliki kesamaan kasus yaitu transaksi skala besar.

DAFTAR PUSTAKA

[1] Ploetz, A., 2018. Seven Nosql Databases In A Week. Birmingham [etc.]: Packt.

[2] Jmeter.apache.org. 2021. Getting Started. [online] Available at: <https://jmeter.apache.org/usermanual/index.html>.

[3] IBM,com. 2021. Difference Between SQL And Nosql . [online] Available at: <https://www.ibm.com/cloud/blog/sql-vs-nosql>.

[4] Academind, 2018. SQL Vs Nosql Or Mysql Vs Mongodb. [video] Available at: <https://www.youtube.com/watch?v=ZS_kXvOeQ5Y&t=14s>.

[5] Udemy Tech, 2018. How To Choose The Right Database? - Mongodb, Cassandra, Mysql, Hbase - Frank Kane. [video] Available at: <https://www.youtube.com/watch?v=v5e_PasMdXc>.

[6] Pal, R., 2020. Jmeter Beginner. [video] Available at: <https://www.youtube.com/playlist?list=PLhW3qG5bs-L-zox1h3eIL7CZh5zJmci4c> .

[7] StackShare, 2020. *Data Stores Tools*. [online] Available at: <https://stackshare.io/data-stores>.

[8] MongoDB. 2021. *What Is Nosql? Nosql Databases Explained*. [online] Available at: <https://www.mongodb.com/nosql-explained>.

[9] DATAVERSITY. 2015. *Case Study: Cassandra Meets Call of Duty - DATAVERSITY*.[online] Available at: <https://www.dataversity.net/case-study-cassandra-meets-call-of-duty/#>.

[10] MongoDB. 2021. *MongoDB Case Study: foursquare*. [online] Available at: <https://www.mongodb.com/post/15400944604/mongodb-case-study-foursquare> .

[11] Redis Labs. 2021. *Coffee Meets Bagel | Redis Labs*. [online] Available at: <https://redislabs.com/case-studies/coffee-meets-bagel>.

[12] MongoDB. 2021. What is replication in MongoDB?. [online] Available at: <https://www.mongodb.com/basics/replication>.

[13] The Distributed SQL. 2021. *Apache Cassandra: The Truth Behind Tunable Consistency, Lightweight Transactions & Secondary Indexes - The Distributed SQL*. [online] Available at: <https://blog.yugabyte.com/apache-cassandra-lightweight-transactions-secondary-indexes-tunable-consistency/> .

[14] Redis.io. 2021. *Replication – Redis*. [online] Available at: <https://redis.io/topics/replication> .

BIODATA PENULIS

Budiman Akbar R, lahir pada tanggal 25 September 2000 di Bekasi. Penulis merupakan mahasiswa Teknologi Sepuluh Nopember (ITS). Penulis aktif dalam berorganisasi di Himpunan Mahasiswa Teknik Computer-Informatika tahun 2019/2020 dalam departemen Dalam Negeri sebagai staff dan di Himpunan Mahasiswa Teknik Computer-Informatika tahun 2020/2021 dalam departemen Dalam Negeri sebagai wakil ketua departemen.